Форма № 17

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Кафедра програмних засобів

(найменування кафедри)

КУРСОВИЙ ПРОЄКТ (РОБОТА)

з дисципліни «Об'єктно-орієнтоване програмування»
(назва дисципліни)

на тему: «Чат-бот стоматологічних послуг»

Студентів 2 курсу КНТ-219 групи
спеціальності 122 Комп'ютерні науки та інформаційні технології
освітня програма (спеціалізація) системи штучного інтелекту
Потурай Е.Г.
(прізвище та ініціали)
Мєдвєдєв К.В.
(прізвище та ініціали)
Доценко С.С.
(прізвище та ініціали)
Сотер Б.Д.
(прізвище та ініціали)
Керівник професор, к.т.н., Табунщик Г. В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала _____
Кількість балів: _____Оцінка: ECTS _____

Члени комісії _____ Табунщик Г. В.
(підпис) (прізвище та ініціали)
_____ Каплієнко Т. І.
(підпис) (прізвище та ініціали)
_____ Шитікова О.В.
(підпис) (прізвище та ініціали

2020 рік

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ІІРЕ, ФКНТ_____
Кафедра програмних засобів_____
Ступінь вищої освіти ___бакалавр_____
Спеціальність 122 Комп'ютерні науки та інформаційні технології
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерні науки _____
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
_____С.О. Субботін
“____” _____20___року

З А В Д А Н Н Я

НА КУРСОВИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТІВ

Потураю Едуарду Григоровичу, Мєдвєдєву Кирилу Віталійовичу, Доценку Степану Сергійовичу, Сотеру Богдану Дмитровичу
(прізвище, ім’я, по батькові)

1. Тема проєкту (роботи) Чат-бот стоматологічних послуг
керівник проєкту (роботи) Табунщик Галина Володимирівна, к.т.н., професор ,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від _____
2. Строк подання студентом проєкту (роботи) 20 грудня 2020 року_____
3. Вихідні дані до проєкту (роботи) створити Чат-бот для запису до дитячої стоматології
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області; 2. Аналіз програмних засобів; 3. Основні рішення з реалізації компонентів системи; 4. Керівництво програміста; 5. Керівництво користувача; 6. Додатки.
_____
5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)_____Слайди презентації_____

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Табунщик Г. В., професор		

7. Дата видачі завдання__11 вересня 2020 р._____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів курсового проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1.	Аналіз індивідуального завдання.	1 тиждень	
2.	Аналіз програмних засобів, що будуть використовуватись в роботі.	2 тиждень	
3.	Аналіз структур даних, що необхідно використати в курсової роботі.	3 тиждень	
4.	Затвердження завдання	4 тиждень	
5.	Вивчення можливостей програмної реалізації структур даних та інтерфейсу користувача.	5-9 тиждень	
6.	Розробка програмного забезпечення	9-13 тиждень	
7.	Проміжний контроль	10 тиждень	Розділи 1-2 ПЗ
8.	Оформлення, відповідних пунктів пояснювальної записки.	10-14 тиждень	Розділи 1-5 ПЗ
9.	Захист курсової роботи.	15 тиждень	

Студент _____ ___Потурай Е.Г.____
(підпис) (прізвище та ініціали)

Студент _____ _____Мєдвєдєв К.В._____
(підпис) (прізвище та ініціали)

Студент _____ _____Сотер Б.Д._____
(підпис) (прізвище та ініціали)

Студент _____ _____Доценко С.С._____
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи) _____ _____Табунщик Г. В.___
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Об’єкт – процес работи чат-боту.

Предмет – програмні засоби для реалізації декстоп симуляції чат-боту.

Мета роботи – створення програмного забезпечення Чат-боту для запису до дитячої стоматології з використання парадігми ООП.

Для реалізації програмного продукту використовувалася мова програмування C++ та середа розробки Qt Creator.

Здійснено опис прийнятих рішень, реалізованих класів, наведено опис полів та методів реалізованих класів.

У даній роботі було проведено дослідження процесу роботи чат-боту, розглянуто особливості мови програмування C++ у середовищі розробки Qt Creator. Розробка графічного інтерфейсу для користувача реалізована за допомогою модуля Qt Designer. Для роботи з даними використовувалися файл формату JSON та база даних SQlite, яка керувалась за допомогою DB Browser (SQlite).

ЧАТ-БОТ, C++, QT CREATOR, SQLITE, JSON, ООП, КЛАСИ, АЛГОРИТМИ, СОРТУВАННЯ, БД, ІНТЕРФЕЙС.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних.

ООП – об’єктно-орієнтоване програмування.

Qt Creator – середовище розробки.

JSON – JavaScript Object Notation/

Інтерфейс – засіб зручної взаємодії користувача з інформаційною системою.

GUI (Graphical User Interface) – графічний інтерфейс користувача.

СУБД – система управління базами даних.

ПІБ – прізвище ім’я по-батькові.

SQL – Structured Query Language.

ВСТУП

Чат-бот – це програма, яка імітує реальну розмову з користувачем. Чат-боти дозволяють спілкуватися за допомогою текстових або аудіо повідомлень на сайтах, в месенджерах, мобільних додатках або по телефону [1].

Використання чат-ботів є ефективним рішенням для бізнесу. Вони мають ряд переваг:

- ефективна взаємодія з клієнтами;
- економність;
- легкість в експлуатації;
- збір відгуків від клієнтів;
- збір інформації про вподобання клієнтів.

Область використання програми включає в себе різні сфери людської діяльності, такі як: E-commerce, інформаційна, освітня, розважальна та, навіть, медична.

Отже, дана курсова робота є актуальною на сьогоднішній день, тому що робота чат-ботів допомогає заощадити час, підвищити ефективність і прибутковість для бізнесу .

Метою даної курсової роботи є затвердження теоретичних основ та практичних аспектів об'єктно-орієнтованого програмування, створення чат-боту для автоматизації роботи з клієнтами стоматологічної клініки.

У роботі необхідно виконати наступні завдання:

- аналіз предметної області:
- розробити відповідні структури даних;
- створити візуальний інтерфейс;
- розробити програму;
- провести тестування;

- оформити пояснювальну записку.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

В даному розділі розглянуті існуючі методи вирішення завдання чат-ботів, запису на прийом та програми, що реалізують чат-ботів.

1.1 Аналіз видів та функцій чат-ботів, як основа предметної області.

Чат-боти можна розділити на 3 основних види:

- Кнопковий бот: він виглядає як кнопки з варіантами дій. «Спілкування» відбувається через натискання кнопок, а бот реагує на них, як на команди. Такий бот підходить для збору даних, визначення тематики звернень, відповідей на типові питання [2].
- Бот-суфлер: підказує користувачеві як більш точно сформулювати питання. Це дозволяє клієнтові не витрачати зайвий час на обдумування питання, а боту легше це питання обробити.
- Розумний робот: це бот з штучним інтелектом, здатний вести повноцінний діалог з клієнтами. Його відповіді максимально наближені до природної людської мови.

Функцій ботів залежить від галузі застосвання. Наприклад:

1. Логістика. Бот допомагає відстежити замовлення, відповідає на питання про вимоги до вантажу, упаковці та ін.
2. Інтернет магазини. Чат-бот допоможе отримати інформацію про товар, відправити його в кошик, а в подальшому буде тримати вас в курсі всіх акцій.
3. Рітейл. Через чат-бота можна отримати консультацію про товари, акції і спеціальні пропозиції магазину.

4. Їжа. Чат-бот може надати вам інформацію про меню, спеціальних пропозиції або години роботи закладу. Деякі боти навіть приймають замовлення через месенджери.
5. Банки. Чат-бот в банку здатний нагадувати про терміни погашення кредиту, конвертувати валюту і виконувати інші корисні функції.

Існує багато методів створення чат-боту. Одним з алгоритмів для програми є:

1. Користувач задає певний запит.
2. Система розділяє весь питання на окремо взяті слова.
3. Над кожним виділеним словом проводиться аналіз.
4. Потім аналізатор зіставляє відповідь з базою даних системи.
5. Після того, як відповідь на запит знайдений в системі, бот показує його користувачеві.

1.2 Огляд існуючих програм та сервісів

В ході написання роботи був проведений аналіз основних програм чат-ботів, який проводився за наступними критеріями:

- спілкування;
- наявність різних функцій;
- простота інтерфейсу.

1.2.1 Чат бот для стоматології

Компанія-видавець: Chatplatforma.

Посилання: https://t.me/StomatClinicBot.

Короткий опис: бот призначений для кліентів стоматології «Stomat». Має такі функції: запис на прийом, послуги, контакти, термінова допомога.

Переваги:

- простий інтерфейс;
- діалог за допомогою кнопок;
- має велику кількість функцій, а саме: запис на прийом та термінова допомога (значить, що тебе приймуть у будь-який робочий час).

Недоліки:

- не виявлено.

Робота програми продемонстрована на рисунку 1.1 та 1.2.

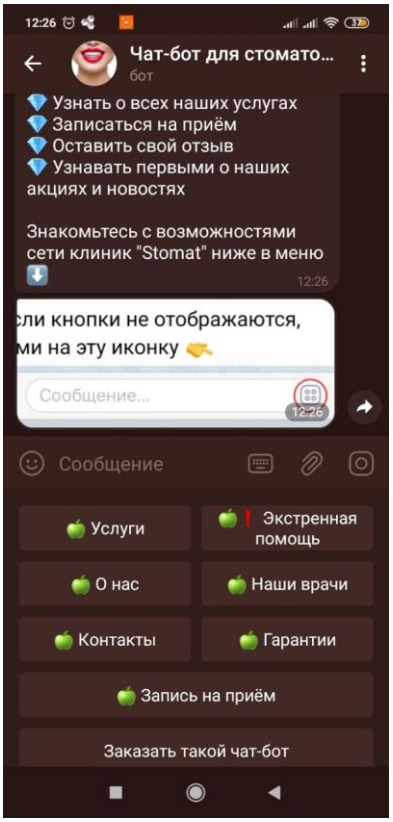


Рисунок 1.1 - Робота програми «Чат бот для стоматології»

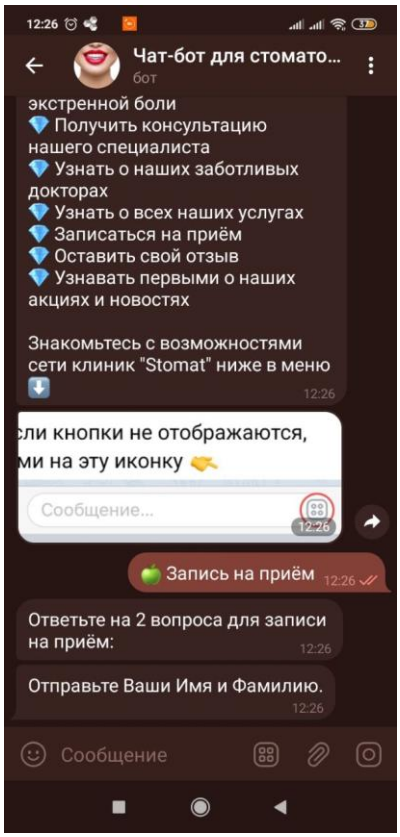


Рисунок 1.2 – Робота програми «Чат бот для стоматології»

1.2.2 Чат-бот Іві

Компанія-видавець: xu.su.

Посилання: https://xu.su/chat-bot-evie.

Короткий опис: призначений для спілкування.

Переваги:

- простий інтерфейс;
- імітує живе спілкування.

Недоліки:

- мало функцій;
- не зрозуміло на які питання може відповісти.

Робота програми продемонстрована на рисунку 1.3.

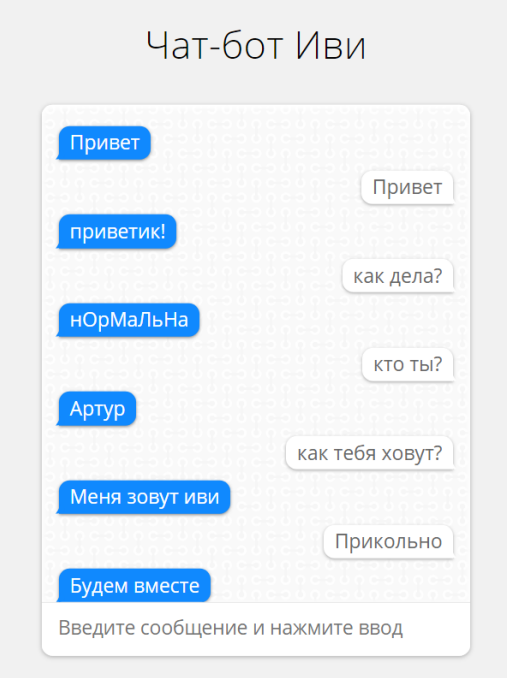


Рисунок 1.3 – Робота програми «Чат бот Іві»

1.2.3 Медсестра Іванка

Компанія-видавець: громадська організація "Антикорупційний штаб".

Посилання: https://t.me/DoctorIvankaBot.

Короткитй опис: завдяки боту можна отримувати інформацію про закупівлі в будь-якій лікарні і залишати скарги на корупцію.

Переваги:

- статистика захворювання на Covid-19 в різних містах;
- можно залишити скаргу на лікарню;
- простий інтерфейс.

Недоліки:

- мало функцій;

– не зрозуміле спілкування.

Робота програми продемонстрована на рисунку 1.4 та 1.5.

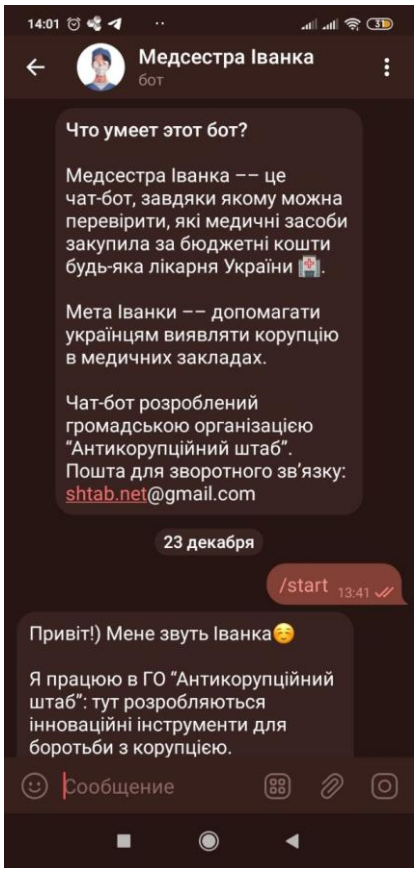


Рисунок 1.4 - Робота програми «Медсестра Іванка»

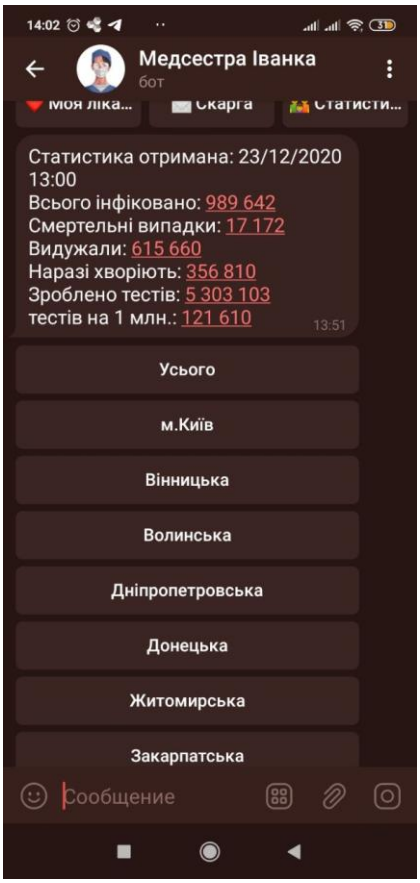


Рисунок 1.5 – Робота програми «Медсестра Іванка»

1.4 Висновки

В цьому розділі було розглянуто основні види, функції та існуючи види чат-ботів.

Отже, існують різні методи реалізації чат-ботів, велика кількість видів з багатьма функціями. Як реалізовувати та який вид обрати залежить лише від програміста.

Наш чат-бот буде мати такі функції:

- запис до лікаря;
- відповідь на поширені питання.

Спілкування буде за допомогою команд та кнопок.

Інтерфейс буде простим.

2 АНАЛІЗ ПРОГРАМНИХ ЗАСОБІВ

В даному розділі розглянуті основні особливості програмних засобів, за допомогою який реалізовано курсовий проект.

2.1 Огляд особливостей мови програмування

C++ — універсальна мова програмування високого рівня з підтримкою декількох парадигм програмування. Зокрема: об’єктно-орієнтованої та процедурної. Розроблена Б’ярном Страуструпом у 1979 році [3].

При створенні C++ прагнули зберегти сумісність з мовою C. Більшість програм на C справно працюватимуть і з компілятором C++. C++ має синтаксис, заснований на синтаксисі C.

Нововведеннями C++ порівняно з C є:

- підтримка об'єктно-орієнтованого програмування через класи;
- підтримка узагальненого програмування через шаблони;
- доповнення до стандартної бібліотеки;
- додаткові типи даних;
- обробка винятків;
- простори імен;
- вбудовані функції;
- перевантаження операторів;
- перевантаження імен функцій;
- посилання і оператори управління вільно розподіленою пам'яттю.

Мова C++ багато в чому є надмножиною C. Нові можливості C++ включають оголошення у вигляді виразів, перетворення типів у вигляді функцій, оператори new і delete, тип bool, посилання, розширене поняття

константності, функції, що підставляються, аргументи за умовчанням, перевизначення, простори імен, класи (включаючи і всі пов'язані з класами можливості, такі як успадкування, функції-члени (методи), віртуальні функції, абстрактні класи і конструктори), перевизначення операторів, шаблони, оператор ::, обробку винятків, динамічну ідентифікацію і багато що інше. Мова C++ також у багатьох випадках строго відноситься до перевірки типів, порівняно з C [3].

2.2 Огляд особливостей обраного середовища розробки

Qt Creator – це повністю інтегроване середовище розробки (IDE), яка надає вам інструменти проектування і розробки складних додатків для безлічі настільних і мобільних платформ [4].

Одним з найголовніших досягнень Qt Creator є те, що він дозволяє команді розробників працювати над проектом на різних платформах з використанням загальних інструментів для розробки і налагодження.

Створення проекту дозволить вам:

- групувати файли разом;
- додати власні кроки збірки;
- включити форми і файли ресурсів;
- вказати налаштування для запускаються.

Ви можете або створити проект з нуля, або імпортувати існуючий проект. Qt Creator генерує всі необхідні файли в залежності від типу створюваного проекту. Наприклад, якщо ви оберете створення додатка з графічним інтерфейсом користувача (GUI), Qt Creator створить порожній .ui файл, який ви можете змінити в інтегрованому Qt Designer.

Qt Creator інтегрований з кроссплатформенную системами автоматизації збирання: qmake і CMake. Також ви можете імпортувати існуючі проекти, які не використовують qmake або CMake, і вказати Qt Creator просто проігнорувати вашу систему збирання.

Qt Creator поставляється з редактором коду і Qt Designer для проектування і складання графічних інтерфейсів користувача (GUI) з віджетів Qt.

Ви можете використовувати Qt Designer щоб мати у своєму розпорядженні і налаштовувати ваші віджети або діалоги і тестувати їх використовуючи різні стилі і дозволу екрану. Створені за допомогою Qt Designer віджети і форми легко інтегруються в програмний код з використанням механізму сигналів і слотів Qt, які дозволять вам легко визначити поведінку графічних елементів. Всі властивості, встановлені в Qt Designer, можуть бути динамічно змінені в коді. Більш того, такі особливості як просування віджетів і власні модулі дозволять вам використовувати власні віджети з Qt Designer.

2.3 Огляд класової ієрархії

При розробці програмного забеспечення використовувались такі класи Qt:

1. Для роботи з текстовими даними:
   - QString;
   - QMap.
2. Для роботи з файлом JSON:
   - QJsonDocument;
   - QJsonObject;

- QJsonArray;
- QJsonValue;
- QObject;
- QFile.

3. Для роботи з БД:
   - QSqlDatabase;
   - QSqlQuery;
   - QSqlError.
4. Для роботи з датою ча часом:
   - QDateTime;
   - QTime.
5. Для графічного інтерфейсу:
   - QMessageBox;
   - QPixmap;
   - QMainWindow;
   - QWidget;
   - QLabel;
   - QSplashScreen;
   - QPainter;
   - QDebug.
6. Для роботи з пам’яттю:
   - new;
7. Для обробки помилок:
   - QException;

QMap<Key, T> – надає словник (асоціативну множину), який зберігає відповідності ключів типу Key і значень типу T. QMap зберігає дані, впорядковані по ключу [5].

Клас QSqlQuery забезпечує інтерфейс для виконання SQL запитів і навігації по результуючої вибіркою [5].

Клас QWidget є базовим для всіх об'єктів користувальницького інтерфейсу.

Віджет – це елементарний об'єкт призначеного для користувача інтерфейсу: він отримує події миші, клавіатури та інші події від віконної системи і малює своє зображення на екрані. Кожен віджет має прямокутну форму, і всі вони відсортовані в порядку накладення (Z-order). Віджет обмежений своїм батьком і іншими віджетами, розташованими перед ним [5].

Віджет, що не вбудований в батьківський віджет, називається вікном. Зазвичай вікно має рамку і смугу з заголовком, хоча використовуючи відповідні прапори вікна можна створити вікно без такого зовнішнього оформлення. В Qt QMainWindow і різноманітні підкласи QDialog є найбільш поширеними типами вікон.

3 ОСНОВНІ РІШЕННЯ З РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

В даному розділі розглянуті основні рішення з розробки класів, основні розроблені алгоритми, рішення щодо розробки інтерфейсу користувача, рішення щодо збереження даних та використання бази даних.

3.1 Основні рішення щодо розроблених класів

Структура програми складається з 5 класів, 4 з яких – користувацькі, 1 – класи форм. Один з користувацьких класів є абстрактним. Текст програми знаходиться в додатку А.

Класову структуру програми зображено на рисунку 3.1.

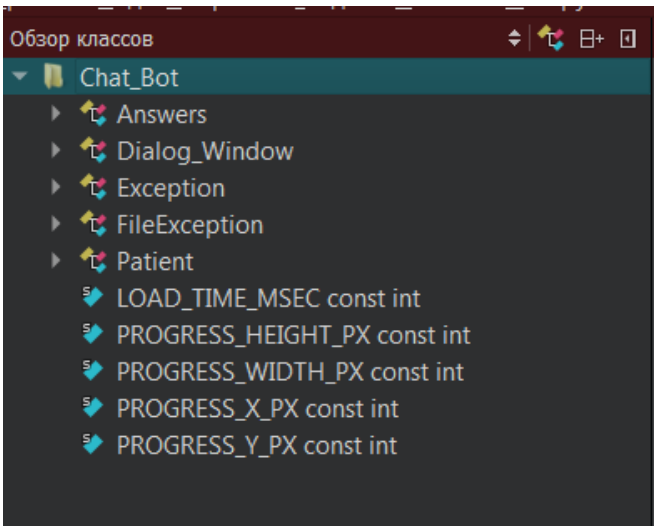


Рисунок 3.1 – Класова структура програми

Класс Answers – клас призначений для роботи з файлом JSON «answers.json» (відкриття, зчитування даних) та пошуку відповіді на питання користувача. Код файлу визначення класу знаходиться в додатку Б. Дані та методи класу наведені у таблиці 3.1.

Таблиця 3.1 – Опис полів та методів класу Answers

Поля та методи класу	Опис
private	
QMap <QString,QString> answer;	Змінна для збереження в key значення полів «key» із файлу json, в value значення полів «value» із файлу json
QString all_content;	Змінна для збереження вмісту файлу «answers.json» без стуктури json
QJsonDocument document;	Змінна для збереження вмісту файлу «answers.json» з структурою json
public	
Answers();	Констуктор
~Answers();	Декструкор
void open_and_read_from_file();	Відкриває файл, зчитує дані, закриває файл.
void read_json_doc();	Перевіряємо правильність структури json, зчитування та запис JSON струтури
void write_to_map();	Зчитування об’єкту (масиву) «map» з файлу, запис струтури json в answer: в key значення полів «key» із файлу json, в value значення полів «value» із файлу json.

Продовження таблиці 3.1

QString return_answer(QString question);	Повертає відповідь на питання користувачеві.

Клас Patient – клас призначений для роботи з даними про записи пацієнтів до лікаря, встановлення зв’язку із БД, розриву цього зв’язку та завантаження інформації з таблиць, що є в наявності у БД, запис даних у таблиці БД. Код файлу визначення класу знаходиться в додатку Б. Дані та методи класу наведені у таблиці 3.2.

Таблиця 3.2 – Опис полів та методів класу Patient

Поля та методи класу	Опис
private	
QString id	Змінна для збереження номеру запису в бд
QString name_child	Змінна для збереження імя дитини
QString number_parent	Змінна для збереження номеру одного з батьків дитини
QString appointment	Змінна для збереження дати та часу запису
QString doctor_specialty	Змінна для збереження спеціальності лікаря
QString doctor_full_name	Змінна для збереження ПІБ доктора
QSqlDatabase db	Змінна для збереження обробки підключення БД
public	
Patient()	Конструктор
~Patient()	Декструктор

Продовження таблиці 3.2

Patient(Patient &A)	Конструктор копіювання
Patient(QString id1, QString name_child1, QString number_parent1, QString appointment1, QString doctor_specialty1, QString doctor_full_name1)	Конструктор з параметрами
void open_db()	Метод призанчений для встановлення звязку з БД.
void input(QString id1, QString name_child1, QString number_parent1, QString appointment1, QString doctor_specialty1, QString doctor_full_name1)	Метод призначений для зміни вмісту полей.
void record_in_db()	Метод призначений для запису даних в таблицю БД «my_patient»
int get_last_id_from_db()	Метод повертає номер останньої записи в таблиці БД «my_patient»
QString *input_available_time(QString specialty,QString status)	Метод повертає вказівник на масив з доступними часами прийому вказаного доктора по спеціальностям. Дані про вільний час завантажується з таблиці «available_time»
void update_status(QString specialty,QString status,QString time)	Метод змінює статус часу прийому на зайнятий в таблиці «available_time»
void sort_db()	Метод сортування таблиці БД «my_patient» за датою прийому (сортування вставками)

Клас Exception – клас для обробки виключних ситуацій, успадкований від класу QException. Код файлу визначення класу знаходиться в додатку Б. Дані та методи класу наведені у таблиці 3.3.

Таблиця 3.3 – Опис полів та методів класу Exception

Поля та методи класу	Опис
private	
QString err	Змінна призначена для зберігання строки з описом проблеми
public	
Exception(QString* s)	Конструктор з параметрами
Exception(*const* char* s)	Конструктор з параметрами
virtual void *what*()	Перевизначений метод, який виводить в QDebug строку з описом проблеми

Клас FileException – абстрактний клас, успадкований від класу Exception. Код файлу визначення класу знаходиться в додатку Б. Дані та методи класу наведені у таблиці 3.4.

Таблиця 3.4 – Опис полів та методів класу FileException

Поля та методи класу	Опис
public	
FileException(QString* s):Exception(s){}	Конструктор з параметрами
FileException(*const* char* s):Exception(s){}	Конструктор з параметрами
void *what*();	Визиває метод класу Exception: what();

Клас Dialog_Window – графічний клас, мітсить методи взаємодії між інтерфейсом, спадкований від класу QMainWindow. Код файлу визначення класу знаходиться в додатку Б. Дані та методи класу наведені у таблиці 3.5.

Таблиця 3.5 – Опис полів та методів класу Dialog_Window

Поля та методи класу	Опис
private	
Ui::Dialog_Window *ui	
Patient My_patient	Об’єкт класу Patient
Answers My_answer	Об’єкт класу Answers
private slots	
void send_message()	Метод призначений для зчитування строки введений користувачем, передача строки в метод displayAnswer(), передача строки в метод Answers: return_answer(), запись відповіді в змінну, передача строки в метод displayMessage ()
void displayMessage(*const* QString &message)	Метод призначений для виводу строки в textEdit
void displayAnswer(*const* QString &message)	Метод призначений для виводу строки в textEdit
void output_about_application()	Метод призначений для виводу QMessageBox
void output_help()	Метод призначений для передачі строки в displayMessage ()
void output_services()	Метод призначений для передачі строки в displayMessage ()

Продовження таблиці 3.5

void start_appointment()	Метод передає строку в displayAnswer(), викликає метод show show _widgets()
void hide_widgets()	Метод ховає віджети, які використовуються для запису на прийом
void show_widgets()	Метод показує віджети, які використовуються для запису на прийом
void change_doctor(*const* QString&)	Метод змінює значення comboBox_doctor на основі вибраного занчення в comboBox_specialty, заповнює значення в comboBox_time з таблиці БД «available_time»
void record_appoinment()	Метод зчитує дані з ведених даних для запису, перевіряє на валідність, якщо дані неправильні, тоді метод завершується, якщо правильні, то змінюється значення в таблиці БД «available_time», дані записуються в змінну класу Patient та викликажться метод Patient :: record_in_db(), для запису в БД, очищуються значення віджетів, передається строка в displayAnswer()
void cancel_record()	Передається строка в displayAnswer()

Продоіження таблиці 3.5

void record_in_txt_tickets(QString name_child, QString number_parent, QString appointment, QString doctor_specialty, QString doctor_full_name);	Відкриває або створює текстовий файл, записує туди дані про прйом.
public	
Dialog_Window(QWidget *parent = *nullptr*)	Конструктор, звязуються сигнали та слоти з віджетами, викликається метод Patient::open_db(), також Answer::open_and_read_from_file(), Answer::read_json_doc(), Answer::write_to_map(), hide_widgets(), заповнюється значення comboBox_specialty, в label вставляються зображення.
~Dialog_Window()	Декструктор, викликається метод Patient::sort_db(),видаляється вікно.

3.2 Основні розроблені алгоритми

В ході написання курсової роботи було написано два алгоритми:

1. Алгоритм введення діалогу з користувачем.
2. Алгоритм сортування БД вставками.

3.2.1 Алгоритм введення діалогу з користувачем

Алгоритм реалізован в класі Dialog_window методом send_message()

Алгоритм приймає текстову строку від користувача, як вихідні дані, виводить її в поле для виведення тексту.

У кінці виконання алгоритм формує вихідні данні: текстову строку від бота, яка виводиться як відповідь.

Вихідні дані мають бути коректними та на українській мові.

Принцип роботи алгоритму зображено на рисунку 3.2.

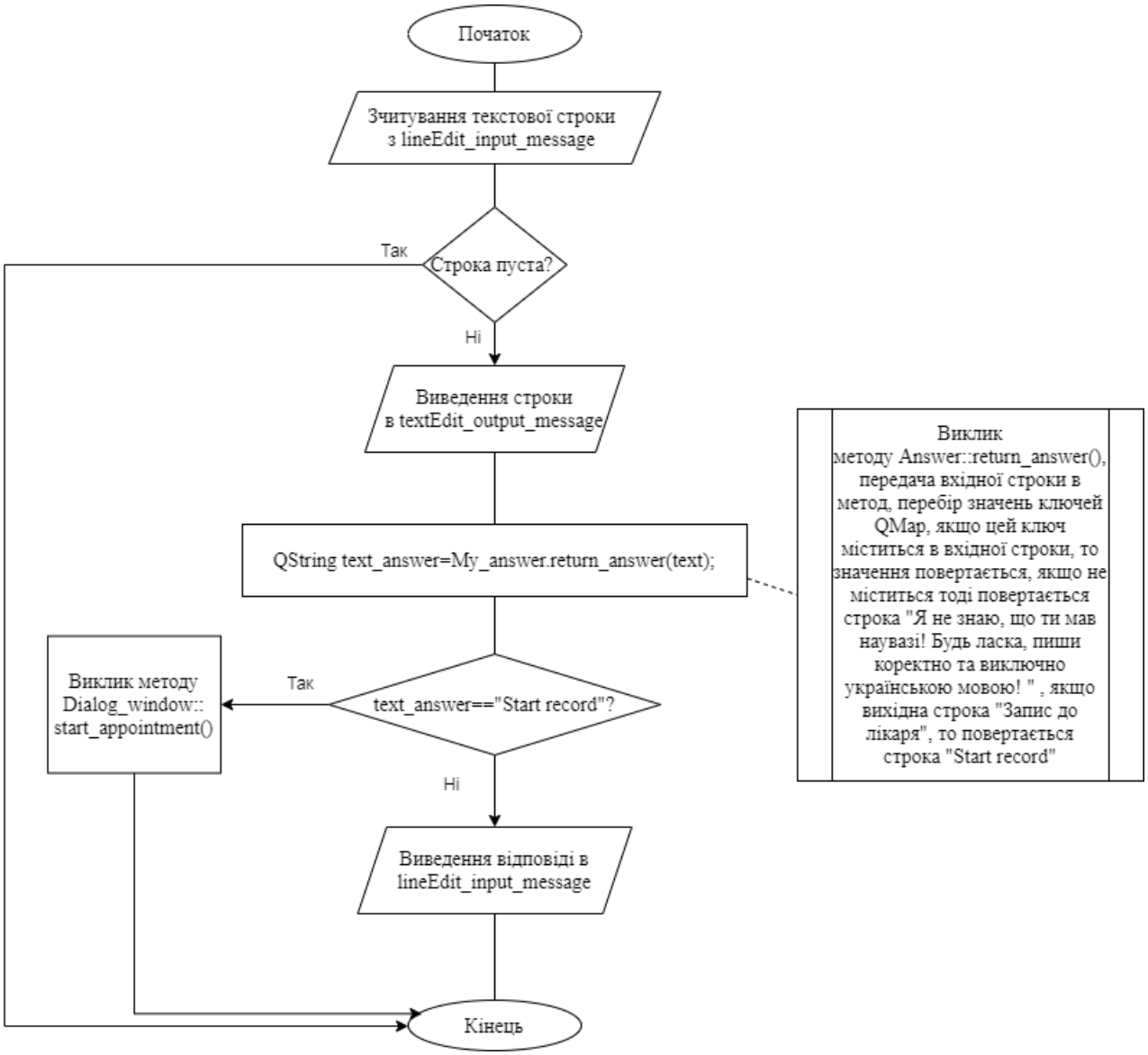


Рисунок 3.2 – Алгоритм введення діалогу з користувачем

3.2.2 Алгоритм сортування БД вставками

Алгоритм реалізован в класі Patient методом sort_db().

Даний алгоритм на основі стандарного алгоритму сортування масиву вставками. Масив даних сортується за датою запису на прийом.

Вхідні дані для алгоритму є дані про пацієнтів з таблиці БД «my_patient».

У кінці виконання алгоритм формує вихідні данні: відсортований масив з даними про записи до лікарів.

Принцип роботи алгоритму зображено на рисунку 3.3.

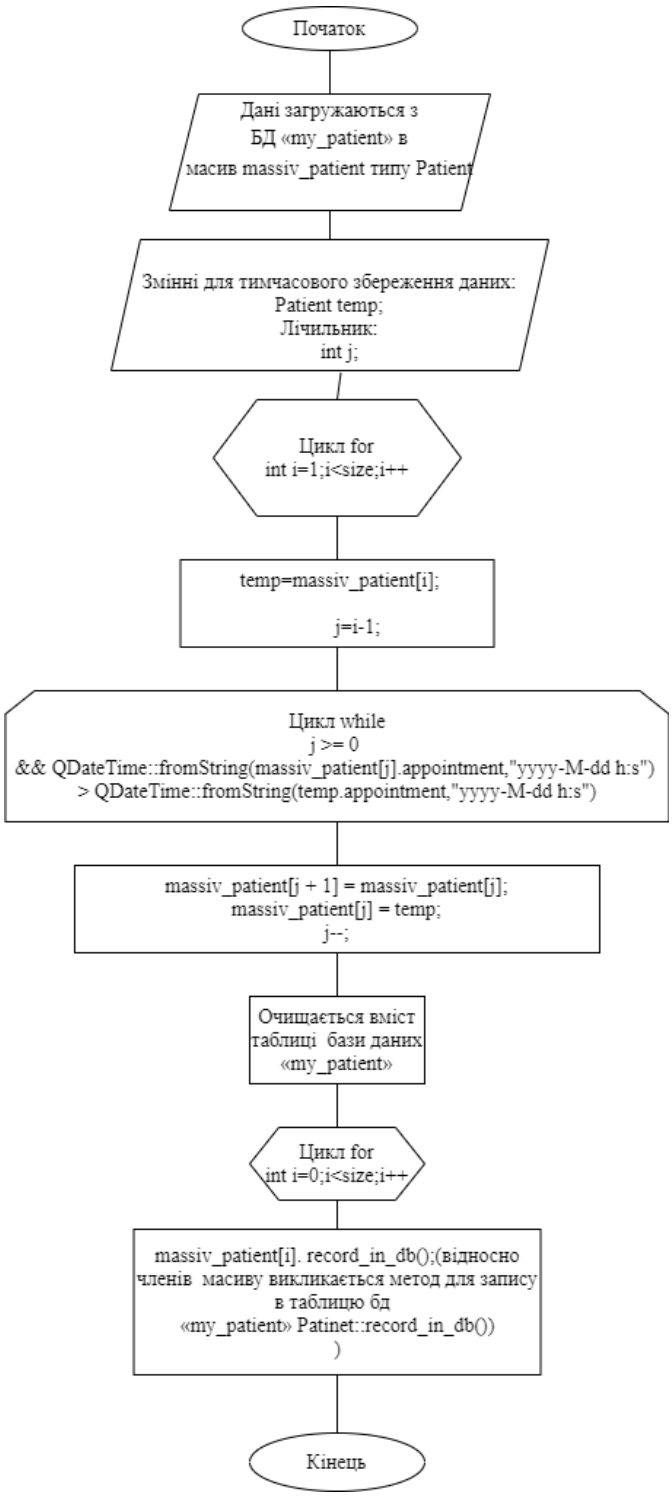


Рисунок 3.3 – Алгоритм сортування БД вставками

3.3 Основні рішення щодо розробки інтерфейсу

Для розробки інтерфейсу програми використовувся вбудований в Qt Creator інструмент для проектування і створення графічних користувацьких інтерфейсів (GUI) з компонентів Qt – Qt Designer.

Під час розробки інтерфейсу користувача було використано наступні елементі графічного інтерфейсу:

- QLabel – для зображення картинок або поля з текстом;
- QTextEdit – для відображення повідомлень;
- QPushButton – кнопки для виконування дій;
- QComboBox – множинний вибір;
- QLineEdit – поле для введення тексту;
- VerticalSpacer – роздільник;
- Layout – компоновщик віджетів;

Структура компонентів відображенна на рисунку 3.4.

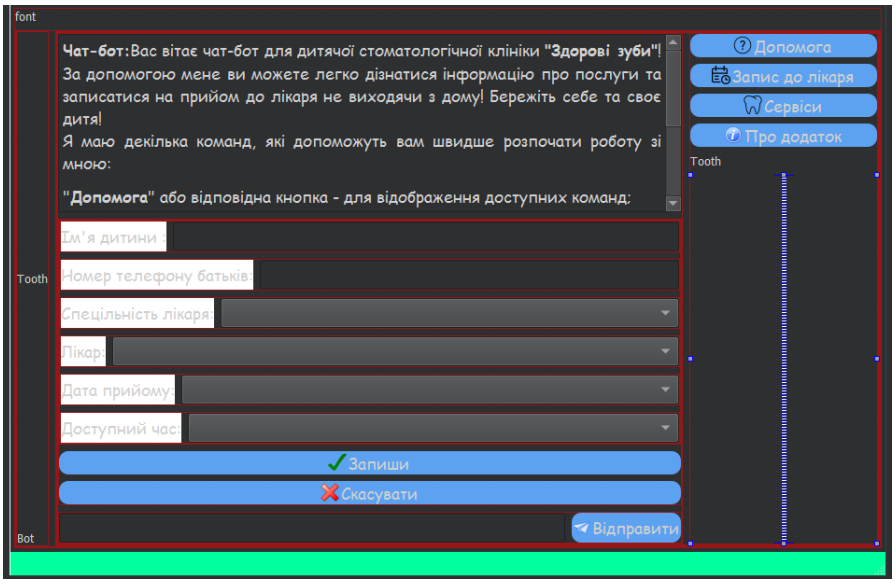


Рисунок 3.4 – Структура компонентів

Для комунікації з користувачем використовується одне вікно з діалогом.

За основу створення інтерфейсу був взятий стандартний вид чату, як приклад інтерфес чат-боту Іві рис.1.3.

Для розтащування елементів використовувся компоновщик. Завдяки кмпоновщику, текстові поля змінюють розмір при змінні розміру вікна.

Для дизайну кнопок та форматування тексту використовувались стилі CSS.

Дизайн вікна доповнений картинками.

Для шрифту був обраний стиль Comic Sans, та був зроблений більший для зручного читання.

При завантаженні головного вікна використовується картинка з завантаженням QSplashScreen. Відображенно на рисунку 3.5.

Інтерфейс головного вікна відображенно на рисунку 3.6.

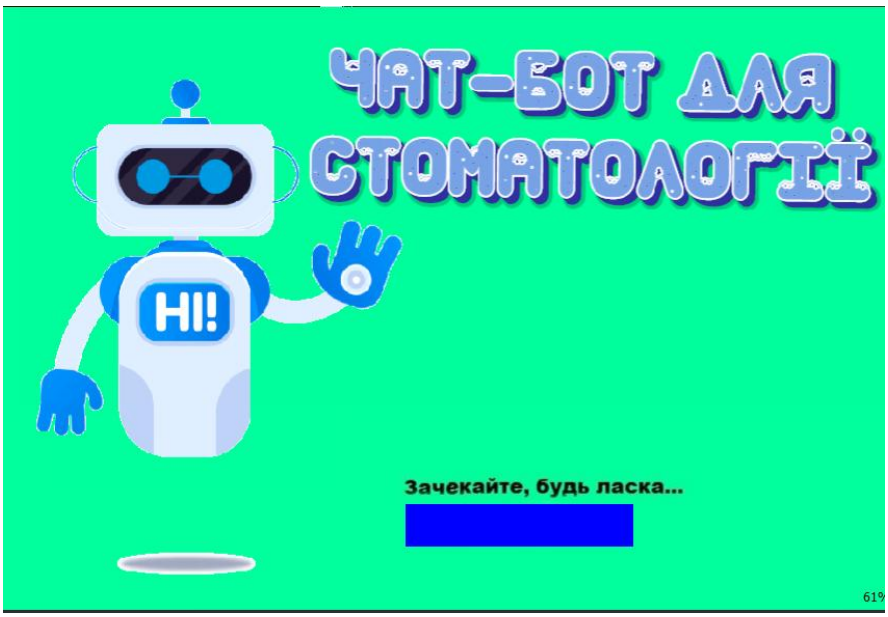


Рисунок 3.5 – Завантаження додатка

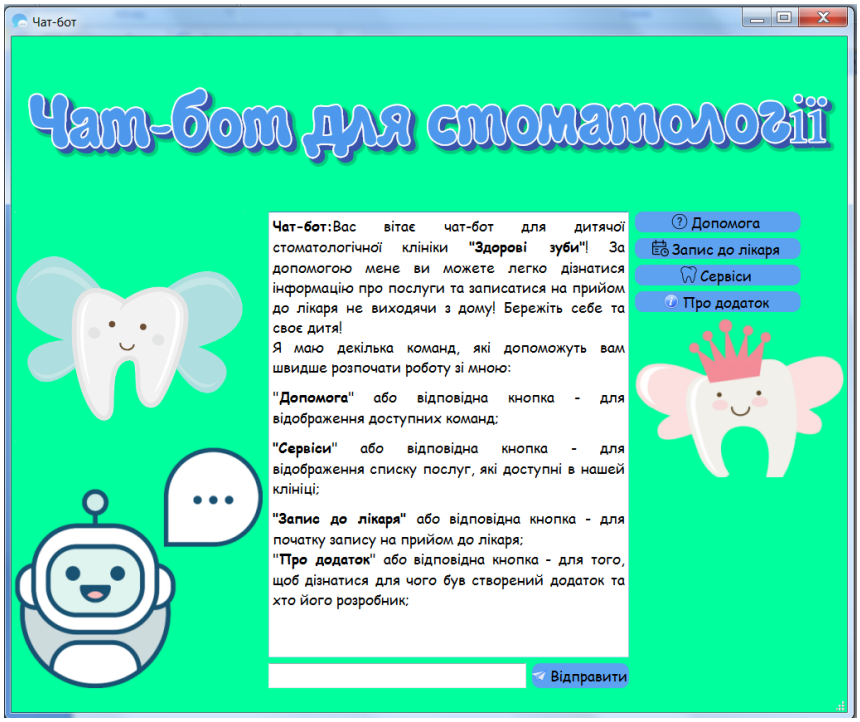


Рисунок 3.6 – Інтерфейс головного вікна

3.4 Основні рішення щодо роботи з даними

Для роботи з даними використувались:

- файл формату JSON “answers.json”;
- база даних SQlite patient.db;
- тексові файли;

Для підключення зображень, які використовуються в інтерфейсі, та файл формату JSON “answers.json” використовується файл ресурсів Qt.

При підключенні БД вказується путь к файлу "../Chat_Bot/DB/patient.db". Папка з файлом зберегається в папці проетку Chat_bot.

Текстові файли використовуються для зберігання талону на прийом. Якщо файлу не існує, то сворюється новий в папці tickets, яка знаходиться в папці проекту Chat_bot. Якщо файл існує, то він відкривається та очищається.

3.4.1 Файл формату JSON “answers.json”

JSON (JavaScript Object Notation) – це формат для зберігання та обміну інформацією, доступною для читання людиною. Файл містить тільки текст і використовує розширення .json.

Файл JSON являє собою більш просту і легку альтернативу розширенню з аналогічними функціями XML (Extensive Markup Language) [6].

Файл містить масив об’єктів «map». Кожен об’єкт містить два ключи:

"key" та "value", та відповідні значення. Приклад на рисунку 3.7.

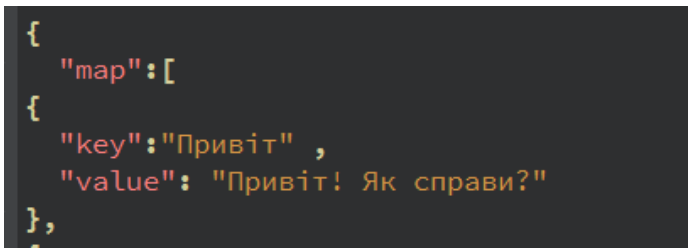


Рисунок 3.7 – Частина вмісту файлу “answers.json”

3.4.2 База даних SQlite patient.db

Для керування базой даних використовувася СУБД – DB Browser (SQlite).

У процесі збереження даних використовуються таблиці, як ключові носії інформації, та представлені наступними екземплярами:

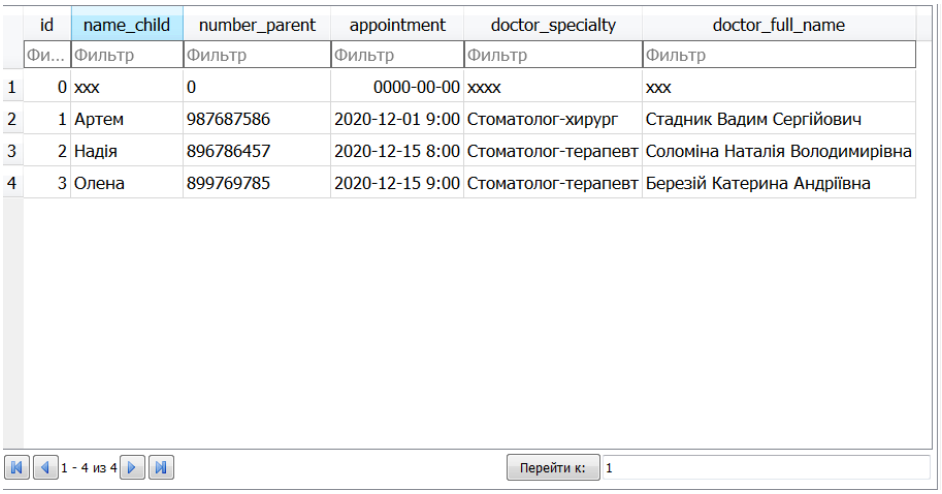
- «my_patient» – таблиця для збереження даних про запис пацієнтів, а саме: ім я дитини, номер одного з батьків, дата та час прийому, спеціальність лікаря, ПІБ лікаря (рис. 3.8). Містить такі поля:

id,name_child,number_parent,appointment,doctor_specialty,doctor_full_name.

- «available_time» – таблиця для збереження даних про доступний час прийомів кожної спеціальності лікарів. Перший стовбець – години, другий – статус (рис. 3.9). Містить такі поля: dentist, status, therapist, status3, surgeon, status2, ortodontist, status4, ortopedist, status1.

Мова для створення запитів SQlite є SQL-подібна, відрізняючись лише синтаксисом (несуттєво) та функціоналом. В ході роботи з базою даних були використані наступні запити для взаємодії:

- CREATE;
- SELECT;
- INSERT;
- DELETE;
- UPDATE.



	id	name_child	number_parent	appointment	doctor_specialty	doctor_full_name
	Фи...	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр
1	0	xxx	0	0000-00-00	xxxx	xxx
2	1	Артем	987687586	2020-12-01 9:00	Стоматолог-хирург	Стадник Вадим Сергійович
3	2	Надія	896786457	2020-12-15 8:00	Стоматолог-терапевт	Соломіна Наталія Володимирівна
4	3	Олена	899769785	2020-12-15 9:00	Стоматолог-терапевт	Березій Катерина Андріївна

1 - 4 из 4 Перейти к: 1

Рисунок 3.8 – Таблиця «my_patient»

	dentist	status	ortopedist	status1	surgeon	status2	therapist	status3	ortodontist	status4
	Фильтр	Филь...	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр
1	8:00	N	10:00	Y	9:00	N	8:00	N	10:00	Y
2	9:00	Y	11:00	Y	10:00	N	9:00	N	11:00	Y
3	10:00	Y	12:00	Y	11:00	Y	10:00	Y	12:00	Y
4	11:00	Y	13:00	Y	12:00	Y	11:00	Y	13:00	Y
5	12:00	Y	14:00	Y	13:00	Y	12:00	Y	14:00	Y
6	13:00	Y	15:00	Y	14:00	Y	13:00	Y	15:00	Y
7	14:00	Y	16:00	Y	15:00	Y	14:00	Y	16:00	Y
8	15:00	Y	17:00	Y	16:00	Y	15:00	Y	17:00	Y

Рисунок 3.9 – Таблиця «my_patient»

3.5 Основні рішення щодо обробки вийняткових ситуацій

Для обробки вийняткових ситуацій використовувалися два класи FileException та Exception.

Оператор try там, де очікується можлива вйняткова ситуація. Виняткова ситуація (строка зі статусом помилки) генерується спеціальним оператором throw з об'єктом-винятком. Оператор catch() перехоплює та оброблює виняткову ситуацію (в QDebug виводиться статус помилки) (рис 4).

```
14:41:52: Запускается D:\kirill\kursovoi\build-Chat_Bot-Desktop_Qt_5_15_0_MinGW_64_bit-Debug\debug\Chat_Bot.exe ...
libpng warning: iCCP: known incorrect sRGB profile
DB file exist
Database loaded successfull!
"no error occurred 0 0"
```

Рисунок 4 – статус помилки в QDebug

Помилки, які обробляються при виконанні програми:

1. Не відкрився файл JSON.
2. Не правильна структура файлу.
3. Не існує файлу БД.
4. Не відкрилась БД.
5. Не обробився SQL-запит.

6. Помилка при роботі з пам’яттю.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

В даному розділі розглянуті призначення, умови застосування, характеристика програми, звертання до програми, початкові та вихідні дані та представлені повідомлення.

4.1 Призначення та умови застосування програми

Програма “Чат-бот для запису до дитячої стоматології” призначена для спілкування з відвідувачів стоматології та запису на прийом до лікаря.

Функції, що виконує програма:

- відповіді на питання;
- запис на прийом;
- інформація про послуги.

До комп’ютеру, на якому виконується програма, висуваються наступні вимоги:

- операційна система Windows 7, 8, 10;
- наявність дисплею;
- наявність миші та клавіатури;
- версія QT creator не нижче 5.

Програма виконана за допомогою мови програмування високого рівня C++ в середовищі розробки QT Creator.

Структура програми складається з 29 файлів:

- Файл проекту - Chat_Bot.pro;
- Заголовні файли: answers.h, exception.h, fileexception.h, dialog_window.h, patient.h;
- Вихідні файли: answers.cpp, exception. cpp, fileexception. cpp, dialog_window. cpp, patient. cpp, main.cpp;
- Файл ресурсів: resource.qrc;
- Файли зображень;
- БД SQLite.

Проект містить наступні класи та їх реалізацію, файли ресурсів, базу даних SQLite. Ієрархія представлена на рисунку 4.1.

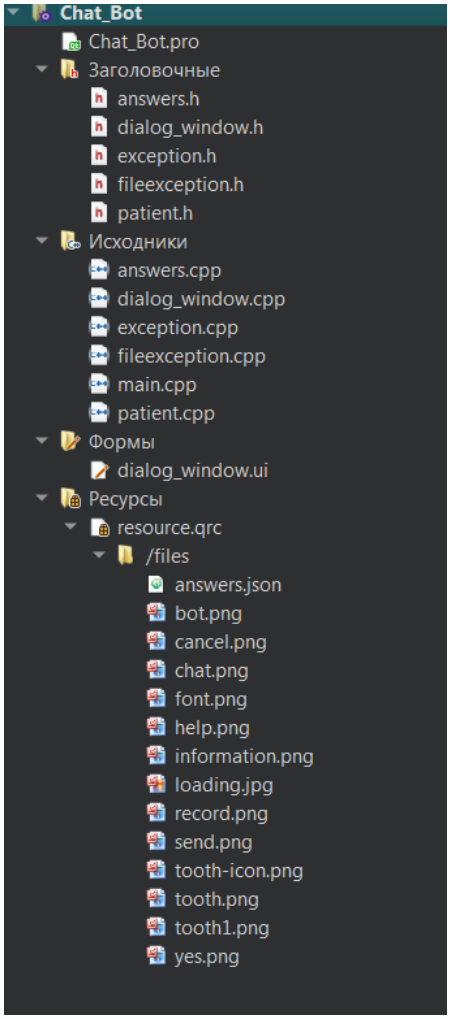


Рисунок 4.1 – Ієрархія файлів проекту

Для запуску програми потрібно попередньо упевнитися в відповідності характеристик комп’ютера, на який встановлена програма, до системних вимог програми. Потрібно упевнитися у наявності всіх файлів програми.

Звертання до програми передбачене двома способами: запустити файл Chat_bot.pro та натиснути «Запуск» (Ctrl+R), або через файл Chat_Bot.exe. Звертання до програми через командний рядок не передбачене, адже програма розроблена саме для роботи через інтерфейс.

Вхідні дані уявляють собою файл формату JSON «answers.json» (з готовим вмістом) та база даних SQlite «patient.db», де міститься дві таблиці: «my_patient» та «available_time».

Вихідні дані уявляють собою файл формату JSON «answers.json» (з готовим вмістом), база даних SQlite «patient.db», де міститься дві таблиці: «my_patient» з відсортованим вмістом та «available_time», текстові файли, де зберігається талон на прийом, якщо при роботі додатка користувач використовав фунцію запис до лікаря.

В ході виконання роботи програми повідомлення виводяться в QDebug, для того щоб запобігти помилок.

5 КЕРІВНИЦТВО КОРИСТУВАЧА

В даному розділі розглянуто призначення програми, умови її виконання, процес виконання програми та повідомлення для користувача.

5.1 Призначення програми

Програма “Чат-бот для запису до дитячої стоматології” призначена для спілкування з відвідувачів стоматології та запису на прийом до лікаря.

5.2 Умови виконання програми

До комп’ютеру, на якому виконується програма, висуваються наступні вимоги:

- наявність дисплею;
- наявність клавіатури та миші;

5.3 Виконання програми

5.3.1 Запуск програми

Для запуску програми потрібно попередньо упевнитися в відповідності характеристик комп’ютера, на який встановлена програма, до системних вимог програми. Другим кроком потрібно упевнитися в наявності всіх бібліотек для роботи програми та в наявності бази даних для повноцінного функціонування.

Звертання до програми передбачене одним способом: через виконуваний файл. Звертання до програми через командний рядок не передбачене, адже програма розроблена саме для візуалізації роботи алгоритмів пошуку найкоротшого шляху.

Після запуску виконуваного файлу Chat_Bot.exe з'являється картинка завантаження прогами, що свідчить про початок роботи з програмою (рис. 3.5).

5.3.2 Виконання роботи з програмою

Спершу користувач має ознайомитися з вступом від бота (рис. 5.1), бот виведе перелік команд та зауваження для подальшого використання.

Бот чутливий к регістру, розуміє лише українську мову.

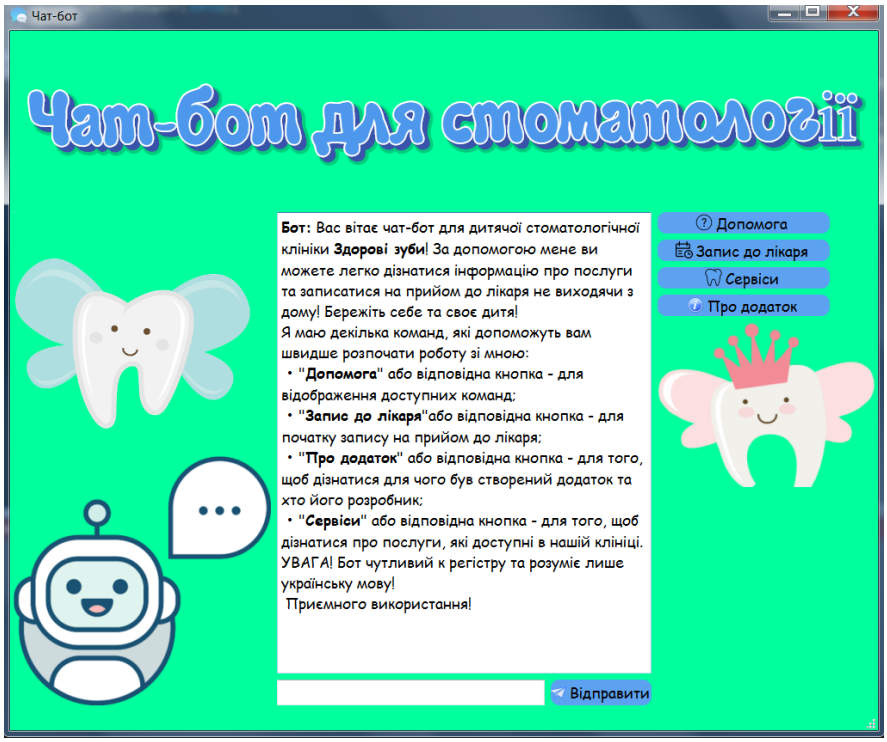


Рисунок 5.1 – Вступ від бота.

Для зручності користувача та для отримання допомоги українською мовою в будь-який момент часу було створено допомогу для програми. Щоб скористатися функцією треба ввести «Допомога» або скористатись відповідною кнопкою. Ознайомитись з переліком команд та запитаннями, на які бот дасть відповідь.

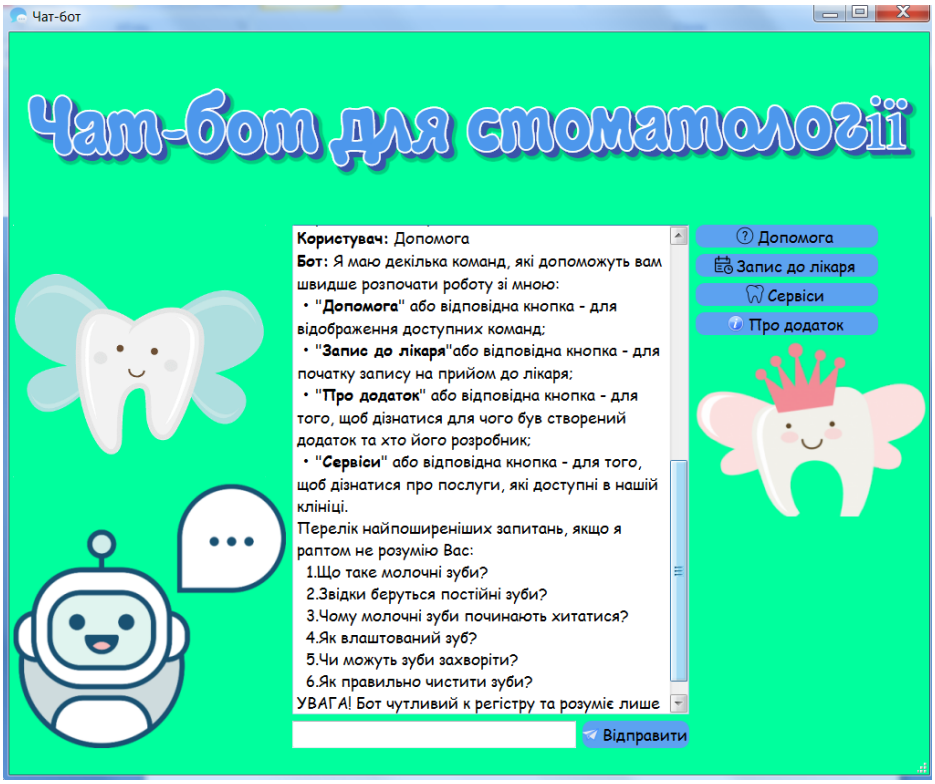


Рисунок 5.2 – Допомога від бота

При використанні функції запису до лікаря зявляються додаткові поля та кнопки.

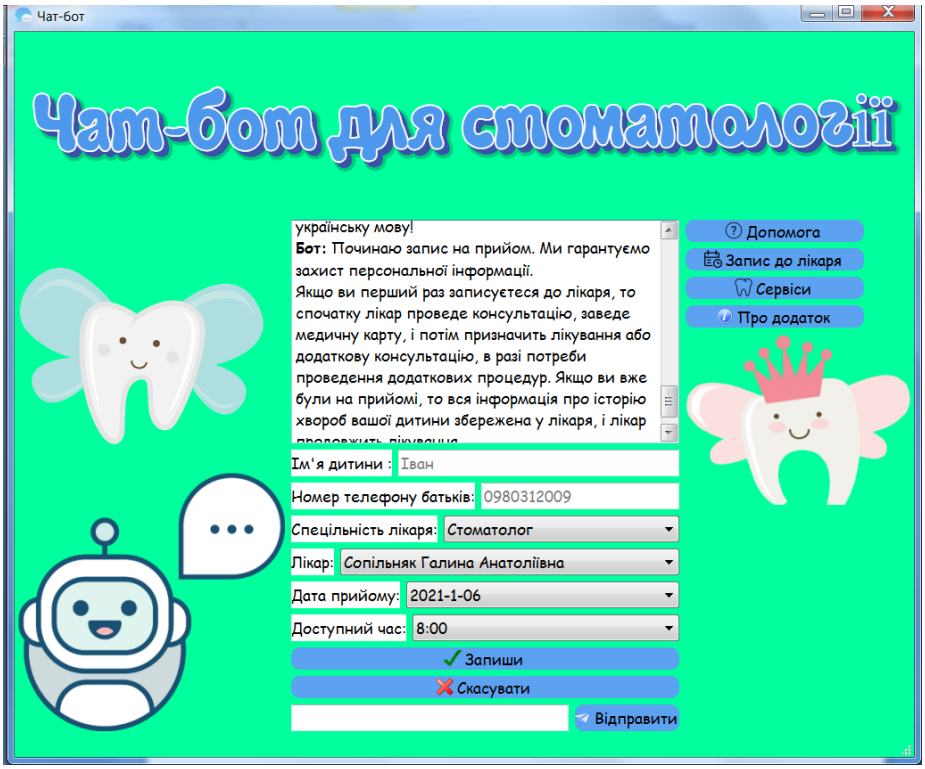


Рисунок 5.3 – Запис до лікаря

Введені дані повинні бути без помилок, або виведеться повідомлення про помилку.

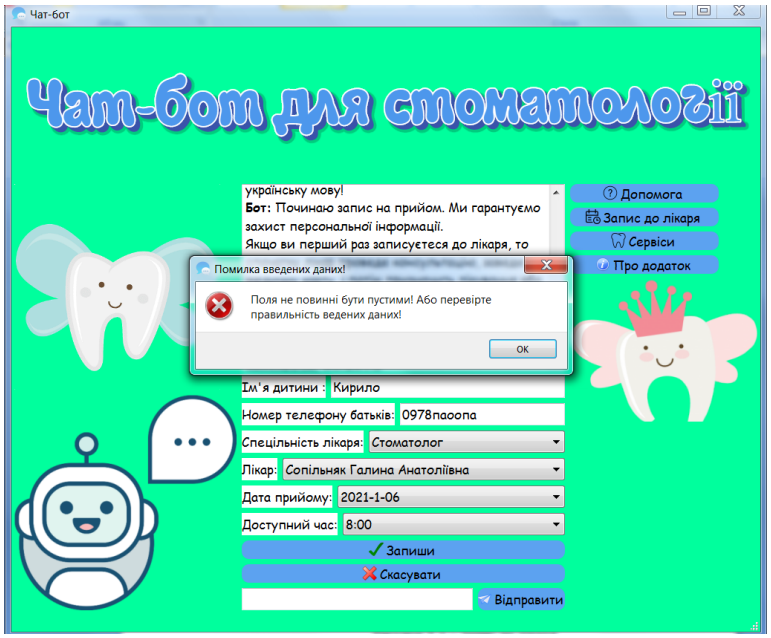


Рисунок 5.4 – Помилка даних

Якщо дані введені правильно бот виконає запис в БД, виведе повідомлення.

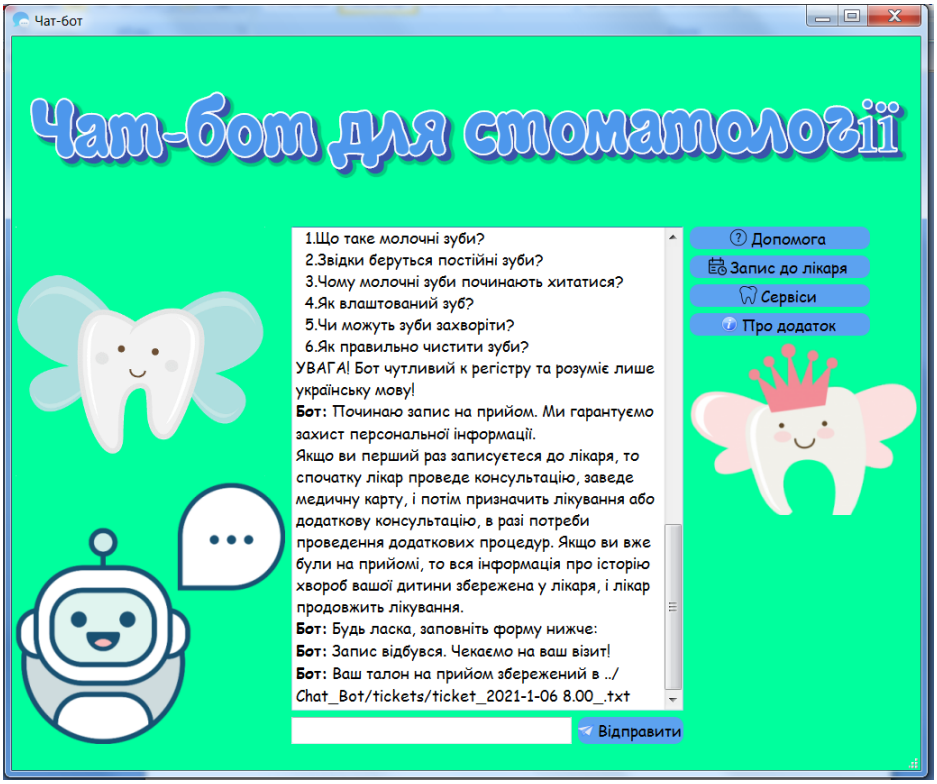


Рисунок 5.5 – Повідомлення після запису

Після цього бот збереже ваш талон на прийом в файлі, путь та назва якого вказано в повідомленні.

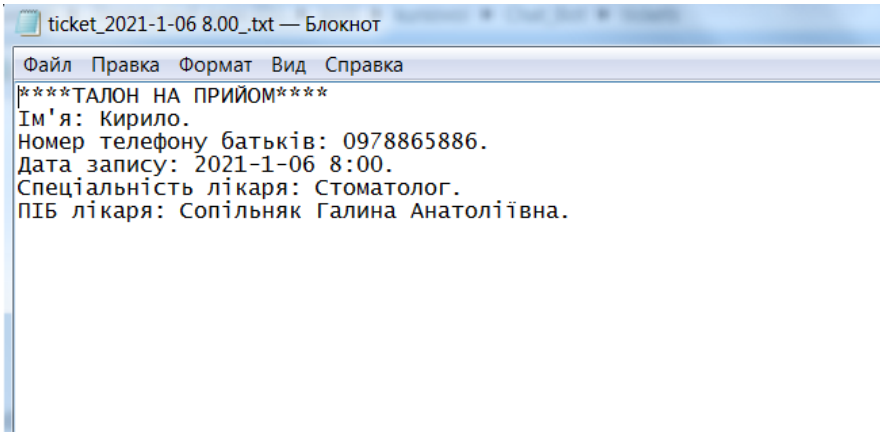


Рисунок 5.6 – Приклад талону

При використанні функції «Сервіси» бот виведе інформацію про послуги.

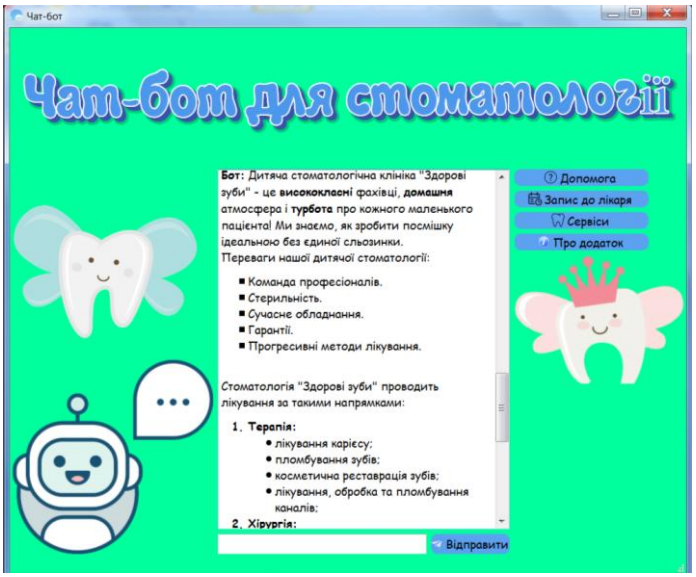


Рисунок 5.6 –Фунція «Сервіси»

Функція «Про додаток» пизначена для того, щоб ознайомитись з авторами проекту.

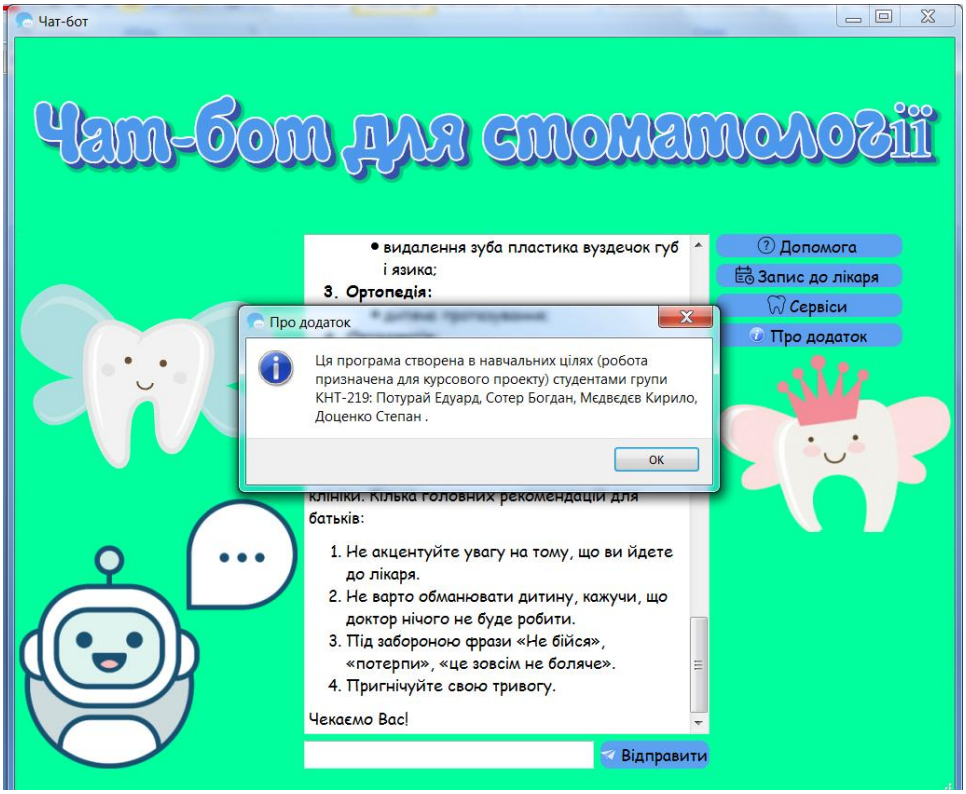


Рисунок 5.7 – Функція «Про додаток»

ВИСНОВКИ

Під час виконання курсового проекту було розроблено програмне забезпечення чат-бот для запису до дитячої стоматології.

Було виконано огляд існуючих методів вирішення завдання та аналогів, проаналізовано переваги та недоліки, розроблено функціонал власного програмного продукту.

Було розроблено алгоритми введення діалогу з користувачем та сортування БД.

Було розроблено максимально простий та зрозумілий інтерфейс, у якому не має зайвих функцій.

Переваги даного розробленого продукту саме в інтерфейсі та алгоритму запису на прийом, який може допомогти автоматизувати роботу стоматології.

Головним недоліком бота є маленький інтервал відповідей на повідомлення.

ПЕРЕЛІК ПОСИЛАНЬ

1. Что такое Чат-Бот: Определение и Руководство [Електрон. ресурс]. – Режим доступу: https://sendpulse.ua/ru/support/glossary/chatbot.

2. Чат-боты для бизнеса. Виды и примеры [Електрон. ресурс]. – Режим доступу: https://webim.ru/blog/13473-chat-bots-for-business-2/#:~:text=Чат-боты%20помогают%20сократить%20издержки,WhatsApp%2C%20Facebook%20Messenger%20и%20т.

3. Огляд і основи мови програмування C++ [Електрон. ресурс]. – Режим доступу: http://www.znannya.org/?view=Cpp_basics.

4. Введение в Qt Creator [Електрон. ресурс]. – Режим доступу: http://doc.crossplatform.ru/qtcreator/2.0.1/creator-overview.html.

5. Qt – Документация на русском [Електрон. ресурс]. – Режим доступу: http://doc.crossplatform.ru/qt/4.7.x/.

6. Что Такое JSON? [Електрон. ресурс]. – Режим доступу: https://www.hostinger.com.ua/rukovodstva/chto-takoe-json/.

ДОДАТОК А ТЕКСТ ПРОГРАМИ

Файл “main.cpp”

```
#include <QApplication>
#include <QLabel>
#include <QSplashScreen>
#include <QPainter>
#include <QTime>
#include "dialog_window.h"

static const int LOAD_TIME_MSEC = 5 * 800;

static const int PROGRESS_X_PX = 397;
static const int PROGRESS_Y_PX = 479;
static const int PROGRESS_WIDTH_PX = 360;
static const int PROGRESS_HEIGHT_PX = 41;

int main( int argc, char* argv[] )
{
   QApplication a( argc, argv );

   QPixmap pix( ":/files/loading.jpg" );
   QSplashScreen splashScreen( pix );
   splashScreen.show();
   a.processEvents();

   QTime time;
   time.start();
   while( time.elapsed() < LOAD_TIME_MSEC )
   {
      const int progress = static_cast< double >( time.elapsed() ) /
LOAD_TIME_MSEC * 100.0;
      splashScreen.showMessage(
         QObject::trUtf8( "%1%" ).arg( progress ),
         Qt::AlignBottom | Qt::AlignRight
      );

      QPainter painter;
      painter.begin( &pix );
```

```
        painter.fillRect(
            PROGRESS_X_PX,
            PROGRESS_Y_PX,
            progress / 100.0 * PROGRESS_WIDTH_PX,
            PROGRESS_HEIGHT_PX, Qt::blue
        );

        painter.end();

        splashScreen.setPixmap( pix );
        a.processEvents();
    }
    splashScreen.hide();

    Dialog_Window w;
      w.show();
      return a.exec();

}
```

Файл “dialog_window.h”

```
#ifndef DIALOG_WINDOW_H
#define DIALOG_WINDOW_H
#include "patient.h"
#include "answers.h"

#include <new>
#include <QMainWindow>

#include <QFile>
#include <QWidget>

#include <QObject>
#include <QMessageBox>
#include <QPixmap>


QT_BEGIN_NAMESPACE
namespace Ui { class Dialog_Window; }
```

```
QT_END_NAMESPACE

class Dialog_Window : public QMainWindow
{
    Q_OBJECT

public:
    Dialog_Window(QWidget *parent = nullptr);
    ~Dialog_Window();

private slots:
    void send_message();
    void displayMessage(const QString &message);
    void displayAnswer(const QString &message);
    void output_about_application();
    void output_help();
    void output_services();
    void start_appointment();
    void hide_widgets();
    void show_widgets();
    void change_doctor(const QString&);
    void record_appoinment();
    void cancel_record();
    void record_in_txt_tickets(QString name_child, QString number_parent, QString
appointment, QString doctor_specialty, QString doctor_full_name);

private:
    Ui::Dialog_Window *ui;
    Patient My_patient;
    Answers My_answer;
};
#endif // DIALOG_WINDOW_H
```

Файл “dialog_window.cpp”

```
#include "dialog_window.h"
#include "ui_dialog_window.h"
#include "exception.h"

Dialog_Window::Dialog_Window(QWidget *parent)
    : QMainWindow(parent)
```

```
    , ui(new Ui::Dialog_Window)
{
    ui->setupUi(this);

    connect(ui->pushButton_send_message, SIGNAL(clicked()), this,
SLOT(send_message()));
    connect(ui->pushButton_output_help, SIGNAL(clicked()),this,
SLOT(output_help()));
    connect(ui->pushButton_output_about_application, SIGNAL(clicked()),this,
SLOT(output_about_application()));
    connect(ui->pushButton_output_services, SIGNAL(clicked()),this,
SLOT(output_services()));
    connect(ui->pushButton_start_appointment, SIGNAL(clicked()),this,
SLOT(start_appointment()));
    connect(ui->comboBox_specialty, SIGNAL(currentIndexChanged(const
QString&)),this, SLOT(change_doctor(const QString&)));
    connect(ui->pushButton_record, SIGNAL(clicked()),this,
SLOT(record_appoinment()));
    connect(ui->pushButton_cancel, SIGNAL(clicked()),this, SLOT(cancel_record()));

    My_patient.open_db();

    My_answer.open_and_read_from_file();
    My_answer.read_json_doc();
    My_answer.write_to_map();

    hide_widgets();

    ui->comboBox_specialty->insertItem(0,"Стоматолог");
    ui->comboBox_specialty->insertItem(1,"Стоматолог-терапевт");
    ui->comboBox_specialty->insertItem(2,"Стоматолог-ортопед");
    ui->comboBox_specialty->insertItem(3,"Стоматолог-хирург");
    ui->comboBox_specialty->insertItem(4,"Стоматолог-ортодонт");

    ui->lineEdit_number->setPlaceholderText("0980312009");
    ui->lineEdit_name->setPlaceholderText("Іван");

    QPixmap pix_bot(":/files/bot.png");

    ui->label_image->setPixmap(pix_bot);
    QPixmap pix_tooth(":/files/tooth.png");
    ui->label_image_tooth->setPixmap(pix_tooth);
    QPixmap pix_tooth1(":/files/tooth1.png");
    ui->label_image_tooth_2->setPixmap(pix_tooth1);
```

```
    QPixmap pix_font(":/files/font.png");
    ui->label_font->setPixmap(pix_font);

    QString start_massage=My_answer.return_answer("Start message");
    displayAnswer(tr("<strong>Бот:</strong> %1").arg(start_massage));




}

Dialog_Window::~Dialog_Window()
{
    My_patient.sort_db();
    delete ui;
}

void Dialog_Window::hide_widgets()
{
    ui->comboBox_date->hide();
    ui->comboBox_doctor->hide();
    ui->comboBox_specialty->hide();
    ui->label_date->hide();
    ui->label_doctor->hide();
    ui->label_name->hide();
    ui->label_number->hide();
    ui->label_specialty->hide();
    ui->lineEdit_name->hide();
    ui->lineEdit_number->hide();
    ui->pushButton_record->hide();
    ui->comboBox_time->hide();
    ui->label_time->hide();
    ui->pushButton_cancel->hide();
}

void Dialog_Window::show_widgets()
{
    ui->comboBox_date->show();
    ui->comboBox_doctor->show();
    ui->comboBox_specialty->show();
    ui->label_date->show();
    ui->label_doctor->show();
    ui->label_name->show();
    ui->label_number->show();
```

```
    ui->label_specialty->show();
    ui->lineEdit_name->show();
    ui->lineEdit_number->show();
    ui->pushButton_record->show();
    ui->comboBox_time->show();
    ui->label_time->show();
    ui->pushButton_cancel->show();
}

void Dialog_Window::displayMessage(const QString &message)
{
    ui->textEdit_output_message->append(message);
}

void Dialog_Window::displayAnswer(const QString &message)
{
    ui->textEdit_output_message->append(message);

}

void Dialog_Window::send_message()
{
    const QString text= ui->lineEdit_input_message->text();


        if(text.isEmpty())
        {
            return;
        }

    displayMessage(tr("<strong>Користувач:</strong> %1").arg(text));

    QString text_answer=My_answer.return_answer(text);

    if(text_answer=="Start record")
    {
        start_appointment();

    }
    else if(text_answer=="About application")
    {
        output_about_application();
    }
    else if(text_answer=="Record")
```

```
    {
        record_appoinment();
    }
    else if(text_answer=="Cancel")
    {
        cancel_record();
    }
    else
    {
    displayAnswer(tr("<strong>Бот:</strong> %1").arg(text_answer));
    }

    ui->lineEdit_input_message->clear();
}

void Dialog_Window::output_help()
{
    QString text_answer=My_answer.return_answer("Допомога");

    displayAnswer(tr("<strong>Бот:</strong> %1").arg(text_answer));
}

void Dialog_Window::output_about_application()
{
    QMessageBox::information(this,"Про додаток","Ця програма створена в
навчальних цілях (робота призначена для курсового проекту) студентами групи
КНТ-219: Потурай Едуард, Сотер Богдан, Мєдвєдєв Кирило, Доценко Степан ."
);
}

void Dialog_Window::output_services()
{
    QString text_answer=My_answer.return_answer("Сервіси");
    displayAnswer(tr("<strong>Бот:</strong> %1").arg(text_answer));
}

void Dialog_Window::start_appointment()
{
    displayAnswer(tr("<strong>Бот:</strong> %1").arg("Починаю запис на прийом.
Ми гарантуємо захист персональної інформації.<br/>"
                                                     " Якщо ви перший раз записуєтеся до лікаря, то
спочатку лікар проведе консультацію, заведе медичну карту, "
                                                     "і потім призначить лікування або додаткову
консультацію, в разі потреби проведення додаткових процедур. "
```

```
                                        "Якщо ви вже були на прийомі, то вся
інформація про історію хвороб вашої дитини збережена у лікаря, і лікар
продовжить лікування."));

   displayAnswer(tr("<strong>Бот:</strong> %1").arg("Будь ласка, заповніть
форму нижче:"));
   show_widgets();
}

void Dialog_Window::change_doctor(const QString&)
{
   QString specialty = ui->comboBox_specialty->currentText();
   QString *time;

   try
   {
      time=new QString[8];
   }
   catch(std::bad_alloc){
      qDebug() << "Allocation falure!";
   }

   if(specialty=="Стоматолог")
   {
      ui->comboBox_doctor->clear();
      ui->comboBox_date->clear();
      ui->comboBox_time->clear();
      ui->comboBox_doctor->insertItem(0,"Сопільняк Галина Анатоліївна");
      ui->comboBox_doctor->insertItem(1,"Березій Юлія Миколаївна");
      ui->comboBox_date-
>insertItem(0,QDate::currentDate().addDays(0).toString("yyyy-M-dd"));
      ui->comboBox_date-
>insertItem(1,QDate::currentDate().addDays(1).toString("yyyy-M-dd"));
      ui->comboBox_date-
>insertItem(2,QDate::currentDate().addDays(2).toString("yyyy-M-dd"));
      ui->comboBox_date-
>insertItem(3,QDate::currentDate().addDays(3).toString("yyyy-M-dd"));
      ui->comboBox_date-
>insertItem(4,QDate::currentDate().addDays(4).toString("yyyy-M-dd"));
      ui->comboBox_date-
>insertItem(5,QDate::currentDate().addDays(5).toString("yyyy-M-dd"));
      time=My_patient.input_available_time("dentist","status");

      for(int i=0;i<8;i++)
```

```
        {
            ui->comboBox_time->insertItem(i,time[i]);
        }
    }
    else if(specialty=="Стоматолог-терапевт")
    {
        ui->comboBox_time->clear();
        ui->comboBox_doctor->clear();
        ui->comboBox_date->clear();
        ui->comboBox_doctor->insertItem(0,"Соломіна Наталія Володимирівна");
        ui->comboBox_doctor->insertItem(1,"Березій Катерина Андріївна");
        ui->comboBox_date-
>insertItem(0,QDate::currentDate().addDays(0).toString("yyyy-M-dd"));
        ui->comboBox_date-
>insertItem(1,QDate::currentDate().addDays(2).toString("yyyy-M-dd"));
        ui->comboBox_date-
>insertItem(2,QDate::currentDate().addDays(4).toString("yyyy-M-dd"));
        ui->comboBox_date-
>insertItem(3,QDate::currentDate().addDays(6).toString("yyyy-M-dd"));
        time=My_patient.input_available_time("therapist","status3");

        for(int i=0;i<8;i++)
        {
            ui->comboBox_time->insertItem(i,time[i]);
        }
    }
    else if(specialty=="Стоматолог-ортопед")
    {
        ui->comboBox_time->clear();
        ui->comboBox_doctor->clear();
        ui->comboBox_date->clear();
        ui->comboBox_doctor->insertItem(0,"Мальков Владислав Віталійович");
        ui->comboBox_doctor->insertItem(1,"Юрченко Олексій Леонідович");
        ui->comboBox_date-
>insertItem(0,QDate::currentDate().addDays(3).toString("yyyy-M-dd"));
        ui->comboBox_date-
>insertItem(1,QDate::currentDate().addDays(6).toString("yyyy-M-dd"));
        ui->comboBox_date-
>insertItem(2,QDate::currentDate().addDays(9).toString("yyyy-M-dd"));
        ui->comboBox_date-
>insertItem(3,QDate::currentDate().addDays(12).toString("yyyy-M-dd"));
        time=My_patient.input_available_time("ortopedist","status1");

        for(int i=0;i<8;i++)
```

```
        {
            ui->comboBox_time->insertItem(i,time[i]);
        }
    }
    else if(specialty=="Стоматолог-хирург")
    {
        ui->comboBox_time->clear();
        ui->comboBox_doctor->clear();
        ui->comboBox_date->clear();
        ui->comboBox_doctor->insertItem(0,"Стадник Вадим Сергійович");
        ui->comboBox_doctor->insertItem(1,"Мірошниченко Ірина Сергіївна");
        ui->comboBox_date-
>insertItem(0,QDate::currentDate().addDays(1).toString("yyyy-M-dd"));
        ui->comboBox_date-
>insertItem(1,QDate::currentDate().addDays(3).toString("yyyy-M-dd"));
        ui->comboBox_date-
>insertItem(2,QDate::currentDate().addDays(5).toString("yyyy-M-dd"));
        ui->comboBox_date-
>insertItem(3,QDate::currentDate().addDays(7).toString("yyyy-M-dd"));
        ui->comboBox_date-
>insertItem(4,QDate::currentDate().addDays(9).toString("yyyy-M-dd"));
        ui->comboBox_date-
>insertItem(5,QDate::currentDate().addDays(11).toString("yyyy-M-dd"));
        time=My_patient.input_available_time("surgeon","status2");

        for(int i=0;i<8;i++)
        {
            ui->comboBox_time->insertItem(i,time[i]);
        }
    }
    else if(specialty=="Стоматолог-ортодонт")
    {
        ui->comboBox_time->clear();
        ui->comboBox_doctor->clear();
        ui->comboBox_date->clear();
        ui->comboBox_doctor->insertItem(0,"Ніколенко Тетяна Анатоліївна");
        ui->comboBox_doctor->insertItem(1,"Дорофєєва Алла Григорівна");
        ui->comboBox_date-
>insertItem(0,QDate::currentDate().addDays(2).toString("yyyy-M-dd"));
        ui->comboBox_date-
>insertItem(1,QDate::currentDate().addDays(4).toString("yyyy-M-dd"));
        ui->comboBox_date-
>insertItem(2,QDate::currentDate().addDays(6).toString("yyyy-M-dd"));
```

```
        ui->comboBox_date-
>insertItem(3,QDate::currentDate().addDays(8).toString("yyyy-M-dd"));
        time=My_patient.input_available_time("ortodontist","status4");

        for(int i=0;i<8;i++)
        {
           ui->comboBox_time->insertItem(i,time[i]);
        }
    }
}

void Dialog_Window::record_appoinment()
{

    QString Sid=QString::number(My_patient.get_last_id_from_db()+1);
    QString Sname_child=ui->lineEdit_name->text();
    QString Snumber_parent=ui->lineEdit_number->text();
    QString Sappointment=ui->comboBox_date->currentText()+" "+ui-
>comboBox_time->currentText();
    QString Sdoctor_specialty=ui->comboBox_specialty->currentText();
    QString Sdoctor_full_name=ui->comboBox_doctor->currentText();
    QString time=ui->comboBox_time->currentText();

    QRegExp rool("[а-яА-Яa-zA-Z]");
    QRegExp rool1("[0-9]");

    if(Sname_child.isEmpty() || Snumber_parent.isEmpty() ||
Snumber_parent.contains(rool) || Sname_child.contains(rool1))
    {
      QMessageBox::critical(this,"Помилка введених даних!","Поля не повинні
бути пустими! Або перевірте правильність ведених даних!");
      return;

    }

    hide_widgets();

    if(Sdoctor_specialty=="Стоматолог")
    {
       My_patient.update_status("dentist","status",time);
    }
    else if(Sdoctor_specialty=="Стоматолог-терапевт")
    {
       My_patient.update_status("therapist","status3",time);
```

```
    }
    else if(Sdoctor_specialty=="Стоматолог-хирург")
    {
        My_patient.update_status("surgeon","status2",time);
    }
    else if(Sdoctor_specialty=="Стоматолог-ортодон")
    {
        My_patient.update_status("ortodontist","status4",time);
    }
    else if(Sdoctor_specialty=="Стоматолог-ортопед")
    {
        My_patient.update_status("ortopedist","status1",time);
    }


My_patient.input(Sid,Sname_child,Snumber_parent,Sappointment,Sdoctor_specialty,
Sdoctor_full_name);
    My_patient.record_in_db();

    ui->lineEdit_name->clear();
    ui->lineEdit_number->clear();

    displayAnswer(tr("<strong>Бот:</strong> %1").arg("Запис відбувся. Чекаємо на
ваш візит!"));


record_in_txt_tickets(Sname_child,Snumber_parent,Sappointment,Sdoctor_specialty,
Sdoctor_full_name);
}

void Dialog_Window::cancel_record()
{
    hide_widgets();
    displayAnswer(tr("<strong>Бот:</strong> %1").arg("Відміна запису!"));
}

void Dialog_Window::record_in_txt_tickets(QString name_child,QString
number_parent,QString appointment,QString doctor_specialty,QString
doctor_full_name)
{
    QString appointment1=appointment;
    appointment1.replace(QRegExp(":"), ".");

    QString file_name="../Chat_Bot/tickets/ticket_" + appointment1 +"_.txt";
```

```
    QFile File(file_name);

    File.open(QIODevice::WriteOnly | QIODevice::Truncate);

    QTextStream out(&File);

    QString inf="****ТАЛОН НА ПРИЙОМ****";
    out<<(inf).toUtf8()<<"\r\n";
    out<<("Ім'я: "+name_child+".").toUtf8()<<"\r\n";
    out<<("Номер телефону батьків: "+number_parent+".").toUtf8()<<"\r\n";
    out<<("Дата запису: "+appointment+".").toUtf8()<<"\r\n";
    out<<("Спеціальність лікаря: "+doctor_specialty+".").toUtf8()<<"\r\n";
    out<<("ПІБ лікаря: "+doctor_full_name+".").toUtf8()<<"\r\n";
    displayAnswer(tr("<strong>Бот:</strong> %1").arg("Ваш талон на прийом
збережений в "+ file_name));
        }
```

Файл “answers.h”

```
#ifndef ANSWERS_H
#define ANSWERS_H
#include <QString>
#include <QMap>

#include <QJsonDocument>
#include <QJsonObject>
#include <QJsonArray>
#include <QJsonValue>

#include <QFile>

#include <QObject>
class Answers
{
    QMap <QString,QString> answer;
    QString all_content;
    QJsonDocument document;

public:
    Answers();
    ~Answers();
```

```
    void open_and_read_from_file();
    void read_json_doc();
    void write_to_map();
    QString return_answer(QString question);
};

        #endif // ANSWERS_H
```

Файл “answers.cpp”

```
#include "answers.h"
#include "exception.h"
#include "fileexception.h"

Answers::Answers()
{
    all_content="";
    answer.clear();
}

Answers::~Answers()
{
    all_content.~QString();
    answer.clear();
    document.~QJsonDocument();
}

void Answers::open_and_read_from_file()
{
    QFile file;
    file.setFileName(":/files/answers.json");

    try
    {
        if(!file.open(QIODevice::ReadOnly | QIODevice::Text))
        {
            QString e = file.errorString();
            throw FileException(&e);
        }
    }
```

```
    catch(FileException &e)
    {
        e.what();
    }

    all_content = file.readAll();
    file.close();
}

void Answers::read_json_doc()
{
    QJsonParseError error;
    document = QJsonDocument::fromJson(all_content.toUtf8(), &error);

    try
    {
        QString eeS = error.errorString();
        int eo = error.offset;
        int ee = error.error;
        eeS = eeS+' '+QString::number(eo)+' '+QString::number(ee);
        throw Exception(&eeS);
    }
    catch(Exception &e)
    {
        e.what();
    }
}

void Answers::write_to_map()
{
    if (document.isObject())
    {
        QJsonObject json = document.object();
        QJsonArray jsonArray = json["map"].toArray();

        foreach (const QJsonValue & value, jsonArray)
        {
            if (value.isObject()){
                QJsonObject obj = value.toObject();
                answer.insert((obj["key"].toString()),obj["value"].toString());
            }
        }
    }
```

```
}

QString Answers::return_answer(QString question)
{
   QMap<QString, QString>::const_iterator i = answer.constBegin();
   QString value;
   int j=answer.size();

   while (i != answer.constEnd())
   {
      if(question.contains(i.key()))
      {
         value=i.value();
         j--;
      }
      else if(j==answer.size() && i!=answer.constBegin())
         value= "Я не знаю, що ти мав наувазі! Будь ласка, пиши коректно та
виключно українською мовою! ";

      ++i;
   }
   return value;
         }
```

Файл “patient.h”

```
#ifndef PATIENT_H
#define PATIENT_H
#include <QString>

#include <QDateTime>
#include <new>

#include <QSqlDatabase>
#include <QSqlQuery>
#include <QSqlError>

#include <QFile>

#include <QDebug>
```

```
class Patient
{
    QString id;
    QString name_child;
    QString number_parent;
    QString appointment;
    QString doctor_specialty;
    QString doctor_full_name;
    QSqlDatabase db;
public:
    Patient();
    ~Patient();
    Patient(Patient &A);
    Patient(QString id1, QString name_child1, QString number_parent1, QString
appointment1, QString doctor_specialty1, QString doctor_full_name1);
    void open_db();
    void input(QString id1, QString name_child1, QString number_parent1, QString
appointment1, QString doctor_specialty1, QString doctor_full_name1);
    void record_in_db();
    int get_last_id_from_db();
    QString *input_available_time(QString specialty,QString status);
    void update_status(QString specialty,QString status,QString time);
    void sort_db();

};

#endif // PATIENT_H
```

Файл “patient.cpp”

```
#include "patient.h"
#include "exception.h"
#include "fileexception.h"

Patient::Patient()
{
    id="0";
    name_child="XXX";
    number_parent="00000000";
    appointment="0-0-0 00:00:00";
    doctor_specialty="XXXX";
```

```
    doctor_full_name="XXXX";
}

Patient::~Patient()
{
    id.~QString();
    name_child.~QString();
    number_parent.~QString();
    appointment.~QString();
    doctor_specialty.~QString();
    doctor_full_name.~QString();

    db.close();
    qDebug() << "Database is closed" << db.connectionNames();
}

Patient::Patient(QString id1, QString name_child1, QString number_parent1, QString
appointment1, QString doctor_specialty1, QString doctor_full_name1)
{
    id=id1;
    name_child=name_child1;
    number_parent=number_parent1;
    appointment=appointment1;
    doctor_specialty=doctor_specialty1;
    doctor_full_name=doctor_full_name1;
}

Patient:: Patient(Patient &A)
{
    id=A.id;
    name_child=A.name_child;
    number_parent=A.number_parent;
    appointment=A.appointment;
    doctor_specialty=A.doctor_specialty;
    doctor_full_name=A.doctor_full_name;
}

void Patient::input(QString id1, QString name_child1, QString number_parent1,
QString appointment1, QString doctor_specialty1, QString doctor_full_name1)
{
    id=id1;
    name_child=name_child1;
    number_parent=number_parent1;
    appointment=appointment1;
```

```
    doctor_specialty=doctor_specialty1;
    doctor_full_name=doctor_full_name1;
}

void Patient::open_db()
{
    db = QSqlDatabase::addDatabase("QSQLITE", "Data");
    db.setDatabaseName("../Chat_Bot/DB/patient.db");

    try
    {
        if(QFile::exists("../Chat_Bot/DB/patient.db"))
            qDebug() << "DB file exist";

        else
            throw FileException("DB file doesn't exists");

        if (!db.open())
        {
            QString dab = db.lastError().text();
            throw FileException(&dab);
        }

        else
            qDebug() << "Database loaded successfull!";
    }
    catch(FileException &e)
    {
        e.what();
    }
}

void Patient::record_in_db()
{
    QSqlQuery query=QSqlQuery(db);

    try
    {
        if(!query.exec("SELECT
id,name_child,number_parent,appointment,doctor_specialty,doctor_full_name
FROM my_patient"))
        {
            QString dbT = query.lastError().databaseText();
            QString dT = query.lastError().driverText();
```

```
            dbT = dbT+' '+dT;
            throw Exception(&dbT);
            return;
        }
    }
    catch(Exception &e)
    {
        e.what();
    }

    query.clear();

    query.prepare("INSERT INTO my_patient
(id,name_child,number_parent,appointment,doctor_specialty,doctor_full_name) "
"VALUES (?, ?, ?,?,?,?)");

    query.addBindValue(id);
    query.addBindValue(name_child);
    query.addBindValue(number_parent);
    query.addBindValue(appointment);
    query.addBindValue(doctor_specialty);
    query.addBindValue(doctor_full_name);

    query.exec();
}

int Patient::get_last_id_from_db()
{
    QSqlQuery query=QSqlQuery(db);
    int last_id;

    try
    {
        if(!query.exec("SELECT id FROM my_patient"))
        {
            QString dbT = query.lastError().databaseText();
            QString dT = query.lastError().driverText();
            dbT = dbT+' '+dT;
            throw Exception(&dbT);
        }
    }
    catch(Exception &e)
    {
        e.what();
```

```
    }

    while (query.next())
    {
            last_id = query.value(0).toInt();
    }
    return last_id;
}

QString* Patient::input_available_time(QString specialty, QString status)
{
    QSqlQuery query=QSqlQuery(db);
    QString *time;

    try
    {
       time=new QString[8];
    }
    catch(std::bad_alloc)
    {
       qDebug() << "Allocation falure!";
    }

    try
    {
       if(!query.exec("SELECT " + specialty + " FROM available_time where
"+status+"='Y'"))
       {
          QString dbT = query.lastError().databaseText();
          QString dT = query.lastError().driverText();
          dbT = dbT+' '+dT;
          throw Exception(&dbT);
          return time=NULL;
       }
    }
    catch(Exception &e)
    {
       e.what();
    }

    int i=0;
    while(query.next())
    { if(i<8)
       {
```

```
            time [i]=query.value(0).toString();
        }
        i++;
    }
    return time;
}


void Patient::update_status(QString specialty, QString status, QString time)
{
    QSqlQuery query=QSqlQuery(db);

    try
    {
        if(!query.exec("UPDATE available_time SET "+ status +" ='N' where "
+specialty+" = '" + time + "'"))
        {
            QString dbT = query.lastError().databaseText();
            QString dT = query.lastError().driverText();
            dbT = dbT+' '+dT;
            throw Exception(&dbT);
        }
    }
    catch(Exception &e)
    {
        e.what();
    }
}

void Patient::sort_db(){
    QSqlQuery query=QSqlQuery(db);

    try
    {
        if(!query.exec("SELECT
id,name_child,number_parent,appointment,doctor_specialty,doctor_full_name
FROM my_patient"))
        {
            QString dbT = query.lastError().databaseText();
            QString dT = query.lastError().driverText();
            dbT = dbT+' '+dT;
            throw Exception(&dbT);
        }
    }
```

```
    catch(Exception &e)
    {
        e.what();
    }

    int size=get_last_id_from_db()+1;

    Patient *massiv_patient=new Patient[size];

    int k=0;

    while(query.next())
    {
        if(k<size)
        {
            massiv_patient[k].id=query.value(0).toString();
            massiv_patient[k].name_child=query.value(1).toString();
            massiv_patient[k].number_parent=query.value(2).toString();
            massiv_patient[k].appointment=query.value(3).toString();
            massiv_patient[k].doctor_specialty=query.value(4).toString();
            massiv_patient[k].doctor_full_name=query.value(5).toString();
        }
        k++;
    }

    Patient temp;
    int j;

    for(int i=1;i<size;i++)
    {
        temp=massiv_patient[i];
        j=i-1;

        while(j >= 0 && QDateTime::fromString(massiv_patient[j].appointment,"yyyy-
M-dd h:s") > QDateTime::fromString(temp.appointment,"yyyy-M-dd h:s"))
                {
                    massiv_patient[j + 1] = massiv_patient[j];
                    massiv_patient[j] = temp;
                    j--;
                }

    }

    for(int i=0;i<size;i++)
```

```
{
  massiv_patient[i].id=QString::number(i);
}

try
{
   if(!query.exec("Delete from my_patient where id>=0"))
   {
      QString dbT = query.lastError().databaseText();
      QString dT = query.lastError().driverText();
      dbT = dbT+' '+dT;
      throw Exception(&dbT);
   }
}
catch(Exception &e)
{
   e.what();
}

query.clear();

for(int i=0;i<size;i++)
{
 massiv_patient[i].db=db;
}

for(int i=0;i<size;i++)
{
 massiv_patient[i]. record_in_db();
}
      }
```

Файл “exception.h”

```
#ifndef EXCEPTION_H
#define EXCEPTION_H

#include <QSqlDatabase>
#include <QFile>

#include <QDebug>
```

```
#include <QString>

#include <QException>

class Exception:public QException
{
   QString err;
public:
   Exception(QString* s);
   Exception(const char* s);
   virtual void what();
};

        #endif // EXCEPTION_H
```

Файл “exception.cpp”

```
#include "exception.h"

Exception::Exception(QString* s)
{
      err=*s;
}

Exception::Exception(const char* s)
{
      err=s;
}

void Exception::what()
{
      qDebug()<<err;
         }
```

Файл “fileexception.h”

```
#ifndef FILEEXCEPTION_H
```

```
#define FILEEXCEPTION_H
#include "exception.h"


class FileException: public Exception
{
public:
   FileException(QString* s):Exception(s){}
   FileException(const char* s):Exception(s){}
   void what();
};

        #endif // FILEEXCEPTION_H
```

Файл “fileexception.cpp”

```
#include "fileexception.h"

void FileException::what()
{
      Exception::what();
}
```